UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/724,616 | 11/28/2000 | Michael L. Ziegler | 10001161-1 | 8427 |

22879 7590 06/07/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

| EXAMINER | |
|---|---|
| NAHAR, QAMRUN | |
| **ART UNIT** | **PAPER NUMBER** |
| 2124 | |

DATE MAILED: 06/07/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/724,616 | ZIEGLER ET AL. | |
| | Examiner | Art Unit | |
| | Qamrun Nahar | 2124 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *15 March 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-14* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-14* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. This action is in response to the appeal brief filed on 3/15/04.

2. The finality of the rejection of the last Office Action is withdrawn in view of new grounds of rejection.

3. Claims 1-14 are pending.

4. Claims 1 and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tsai (U.S. 6,161,196) in view of "GDB Tutorial", by Kierstead et al., 1993 (hereinafter Kierstead).

5. Claims 2-8 and 10-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tsai (U.S. 6,161,196) in view of "GDB Tutorial", by Kierstead et al., 1993 (hereinafter Kierstead), and further in view of Fuchs (U.S. 5,590,277).

### *Claim Rejections - 35 USC § 103*

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1 and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tsai (U.S. 6,161,196) in view of "GDB Tutorial", by Kierstead et al., 1993 (hereinafter Kierstead).

**Per Claim 1:**

Tsai teaches a computer-implemented method for software error recovery ("The invention provides a general error detection and recovery technique that ensures data integrity

for critical data without the need for any modification of source code or executable code, while also providing a high degree of automation and transparency for fault-tolerant configuration and operation." in column 2, lines 32-37); debugging a first set of object code with a first debugger; debugging a second set of object code with a second debugger ("Error detection in the FIG. 7 embodiment is accomplished via replication of a given target program on three different machines, with each copy of the target program controlled by one of the separate backends 28-i on one of the three machines. These backends communicate with the single frontend 100, which serves as a coordinator for the backends 28-i and is operative to determine discrepancies among the different copies of the target program as the copies execute." in column 8, lines 19-29; column 7, lines 1-39; and see Fig. 6, item "GDB"; ***each backend contains a GDB***, which is a debugger that uses information generated by GCC, which is a compiler.); identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code; associating sets of data objects with the checkpoints; automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted ("In operation, each of the copies of the target program may be temporarily stopped at the beginning of the first program instruction. At that point, the user selects the desired voting parameters, e.g. variables to be voted upon and voting times, and recovery parameters, using a creation GUI of the frontend 100. ... only a subset of these or other similar parameters are specified by the user. The frontend 100 then sends the corresponding commands to each of the backends 28-i. The frontend creation GUI described above may be modified in a straightforward manner to allow this user

specification of the voting parameters and recovery parameters. When the appropriate breakpoints have been inserted by the backends 28-i, the execution of all copies of the target program is resumed. For each breakpoint, each backend 28-i will report the value of preselected critical variables to the frontend 100. If the frontend 100 finds that all reported values are identical, then it instructs each backend 28-i to continue execution of its copy of the target program." in column 8, lines 30-48); executing the first set of object code; storing the state information in executing the checkpoint code; and upon detecting an error in execution of the first set of object code, resuming execution of the program using the second set of object code ("FIG. 8 illustrates a situation in which the frontend 100 detects a divergence in the reported values from the backends 28-i. The backend with the minority value is identified as the erroneous backend, and execution of the target program is terminated on the corresponding machine. The erroneous backend in this example is backend2 (28-2). A checkpoint is then taken from one of the non-erroneous backends, e.g., backend 1 (28-1), and that checkpoint data is copied to the machine with the erroneous backend, i.e., backend2 (28-2), and a new target program is initiated on backend2 using the checkpoint data." in column 8, lines 49-61; execution of the program is resumed using the checkpoint data of the non-erroneous backend, e.g., backend 1 (28-1). That is, execution is resumed using the second set of object code of backend 1 (28-1), where each backend inherently contains its own compiler.).

Tsai does not explicitly teach compiling program source code into the first set of object code with a first compiler or compiling the program source code into the second set of object code with a second compiler. However, Kierstead teaches that to invoke the debugger, GDB, the program source code must first be compiled (see for example, "=> Compile the program using

the command: gcc –g gdb_basic.c ... The –g option includes additional information for symbolic debugging.", on pg. 5, see section labeled "4 Using GDB", lines 1-8). Therefore, the program source code must first be compiled at each backend before each GDB can be invoked.

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Tsai to include that compiling program source code into the first set of object code with a first compiler and compiling the program source code into the second set of object code with a second compiler using the teaching of Kierstead. The modification would be obvious because one of ordinary skill in the art would be motivated to use a well-known debugger, GDB, to debug programs.

**Per Claim 9:**

This is an apparatus version of the claimed method discussed above, claim 1, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, this claim is also obvious.

8. Claims 2-8 and 10-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tsai (U.S. 6,161,196) in view of "GDB Tutorial", by Kierstead et al., 1993 (hereinafter Kierstead), and further in view of Fuchs (U.S. 5,590,277).

**Per Claim 2:**

The rejection of claim 1 is incorporated, and further, the combination of Tsai and Kierstead does not explicitly teach upon detecting an error in execution of the first set of object

code, initially re-executing the first set of object code; and resuming execution using the second set of object code if the first set of object code fails in re-execution. Fuchs teaches upon detecting an error in execution of the first set of object code, initially re-executing the first set of object code; and resuming execution using the second set of object code if the first set of object code fails in re-execution (column 15, lines 48-67 to column 16, lines 1-10 and column 17, lines 26-31).

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by the combination of Tsai and Kierstead to include upon detecting an error in execution of the first set of object code, initially re-executing the first set of object code; and resuming execution using the second set of object code if the first set of object code fails in re-execution using the teaching of Fuchs. The modification would be obvious because one of ordinary skill in the art would be motivated to debug the faulty program, that is, to recover the faulty program (Fuchs, column 1, lines 58-67 to column 2, lines 1-32).

**Per Claim 3:**

The rejection of claim 2 is incorporated, and Fuchs further teaches re-executing the first set of object code a selected number of times before resuming execution using the second set of object code (column 17, lines 26-31).

**Per Claim 4:**

The rejection of claim 3 is incorporated, and Fuchs further teaches ceasing resumption of execution of the first and second sets of object code if an error is detected in executing both sets of object code (column 17, lines 31-37).

**Per Claim 5:**

Tsai teaches a computer-implemented method for software error recovery ("The invention provides a general error detection and recovery technique that ensures data integrity for critical data without the need for any modification of source code or executable code, while also providing a high degree of automation and transparency for fault-tolerant configuration and operation." in column 2, lines 32-37); debugging a first set of object code with a first debugger; debugging a second set of object code with a second debugger ("Error detection in the FIG. 7 embodiment is accomplished via replication of a given target program on three different machines, with each copy of the target program controlled by one of the separate backends 28-i on one of the three machines. These backends communicate with the single frontend 100, which serves as a coordinator for the backends 28-i and is operative to determine discrepancies among the different copies of the target program as the copies execute." in column 8, lines 19-29; column 7, lines 1-39; and see Fig. 6, item "GDB"; ***each backend contains a GDB***, which is a debugger that uses information generated by GCC, which is a compiler.); identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code; associating sets of data objects with the checkpoints; automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data

objects for recovery if execution of the program is interrupted ("In operation, each of the copies of the target program may be temporarily stopped at the beginning of the first program instruction. At that point, the user selects the desired voting parameters, e.g. variables to be voted upon and voting times, and recovery parameters, using a creation GUI of the frontend 100. ... only a subset of these or other similar parameters are specified by the user. The frontend 100 then sends the corresponding commands to each of the backends 28-i. The frontend creation GUI described above may be modified in a straightforward manner to allow this user specification of the voting parameters and recovery parameters. When the appropriate breakpoints have been inserted by the backends 28-i, the execution of all copies of the target program is resumed. For each breakpoint, each backend 28-i will report the value of preselected critical variables to the frontend 100. If the frontend 100 finds that all reported values are identical, then it instructs each backend 28-i to continue execution of its copy of the target program." in column 8, lines 30-48); executing the first set of object code; storing the state information in executing the checkpoint code; and upon detecting an error in execution of the first set of object code, selecting the second set of object code in resuming execution of the program ("FIG. 8 illustrates a situation in which the frontend 100 detects a divergence in the reported values from the backends 28-i. The backend with the minority value is identified as the erroneous backend, and execution of the target program is terminated on the corresponding machine. The erroneous backend in this example is backend2 (28-2). A checkpoint is then taken from one of the non-erroneous backends, e.g., backend 1 (28-1), and that checkpoint data is copied to the machine with the erroneous backend, i.e., backend2 (28-2), and a new target program is initiated on backend2 using the checkpoint data." in column 8, lines 49-61).

Tsai does not explicitly teach compiling program source code into the first set of object code with a first compiler or compiling the program source code into the second set of object code with a second compiler or selecting the first set of object code in resuming execution of the program.

However, Kierstead teaches that to invoke the debugger, GDB, the program source code must first be compiled (see for example, "=> Compile the program using the command: gcc –g gdb_basic.c ... The –g option includes additional information for symbolic debugging.", on pg. 5, see section labeled "4 Using GDB", lines 1-8). Therefore, the program source code must first be compiled at each backend before each GDB can be invoked.

Fuchs teaches selecting the first set of object code in resuming execution of the program (column 15, lines 48-67 to column 16, lines 1-10).

It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Tsai to include that compiling program source code into the first set of object code with a first compiler; compiling the program source code into the second set of object code with a second compiler; and selecting the first set of object code in resuming execution of the program using the teaching of the combination of Kierstead and Fuchs. The modification would be obvious because one of ordinary skill in the art would be motivated to debug the faulty program, that is, to recover the faulty program (Fuchs, column 1, lines 58-67 to column 2, lines 1-32).

**Per Claims 6-8:**

These are another versions of the claimed method discussed above (claims 2-4, respectively), wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also obvious.

**Per Claim 10:**

This is an apparatus version of the claimed method discussed above, claim 5, wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, this claim is also obvious.

**Per Claims 11-14:**

These are computer program product versions of the claimed method discussed above (claims 5-8, respectively), wherein all claim limitations also have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also obvious.

***Response to Arguments***

9. In view of the appeal brief filed on 3/15/04, PROSECUTION IS HEREBY REOPENED. New grounds of rejection are set forth above.

To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) request reinstatement of the appeal.

If reinstatement of the appeal is requested, such request must be accompanied by a supplemental appeal brief, but no new amendments, affidavits (37 CFR 1.130, 1.131 or 1.132) or other evidence are permitted. See 37 CFR 1.193(b)(2).

10. Applicant's arguments filed on 3/15/04 have been considered but are moot in view of the new ground(s) of rejection.

*In the remarks, the applicant argues that:*

a) The Office Actions are mistaken in the allegation that Tsai's backends inherently contain compilers. As summarized above, in the present invention the program source code is compiled with a first compiler into a first set of object code and compiled with a second compiler into a second set of object code. As explained below, the two compilers are not inherent in Tsai's backends.

MPEP 2112 summarizes the requirements to establish inherency:

...

The Office Actions have not shown that Tsai's backends necessarily include separate compilers that generate separate object code sets.

The most recent Office Action points, to the GDB element in backend 28 in Tsai's FIG. 6 in support of the inherency allegation. Tsai explains that the GDB is a debugger process, which controls debugger operations undertaken by the backend. (col. *7,11. 1-15).* A debugger is generally capable of performing many low-level tasks such as managing breakpoints, executing debugger commands when breakpoints are encountered, and printing and modifying the values

of variables. (col. 2, 1. 66 - col. 5, 1. 2). The Office Action is mistaken in reasoning that, each backend must inherently contain "a GCC, a compiler, for the GBD [sic], for the debugger to work."

Even though the Office Actions do not explain and no evidence is provided to describe a GCC, it is assumed that GCC refers to the GNU Compiler Collection (GCC). The website, http://gec.gnu.org/ explains that the; GCC contains front ends for C, C++, ObjectiveC, Fortran, Java, and Ada, as well as libraries for these languages (emphasis added; see APPENDIX SHOWING GCC WEBSITE in this brief). This clearly indicates that a GCC is a **front** end tool, not a back end component. Furthermore, Tsai clearly shows GDB as a backend tool, and the **APPENDIX SHOWING GDB USAGE** (also in this brief) indicates that the GDB requires information generated from a compiler. There is no suggestion that the GCC is inherently part of a back end GDB. Indeed, the suggestion is that the front ends and back ends are separate.

*Examiner's response:*

a) Claims 1 and 9 are now rejected under 35 U.S.C. 103(a) as being unpatentable over Tsai (U.S. 6,161,196) in view of "GDB Tutorial", by Kierstead et al., 1993 (hereinafter Kierstead). The combination of Tsai and Kierstead clearly shows each and every limitation in claims 1 and 9. See the rejection above in paragraph 7 for rejection to claims 1 and 9.

*In the remarks, the applicant argues that:*

b) As explained in the response to the first Office Action, Tsai may use one compiler to generate object code and then install and run copies of the object code on the various backends.

Furthermore, Tsai's description appears to suggest that copies of a program are run (col. 9, 1. 64 - cot. 10, 1. 8), in which case each copy would be from one compiler.

Therefore, the Office Actions fail to show that Tsai's backends inherently include separate compilers that generate separate object code sets.

The claims include further limitations of resuming execution of the program using the second set of object code upon detecting an error in execution of the first set of object code. The Office Actions do not show that these limitations are taught in Tsai's col. 8, 11. 49-61. As this section of Tsai clearly states, the target program on the machine with the erroneous backend is terminated and a new target program is initiated on the machine using checkpoint data. As Tsai later explains, the newly started program is not from a second compiler. Tsai's newly started program is simply a copy of the target program (cot. 9, 1. 64 - cot. 10, 1. 8). Thus, Tsai does not resume execution using a second set of object code as claimed.

*Examiner's response:*

b) In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., separate compilers) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Claims 1 and 9 recite a first compiler and a second compiler. The first compiler and second compiler do not necessarily have to be two different compilers; the same compiler could be used twice.

*In the remarks, the applicant argues that:*

c) The claims of group II include the limitations of claim 1 and further limitations related to selecting between the first and second sets of object code in resuming execution. The Office Actions fail to show a teaching of these limitations by either Tsai or Fuchs. The explanation provided in the Office Actions is: Fuchs teaches selecting the first set of object code in resuming execution of the program (column 15, lines 48-67 to column 16, lines 1-10), and Tsai teaches selecting the second set of object code (col. 8. 11. 49-61). It is respectfully submitted that the Office Action appears to ignore the claimed aspect of selecting between the first and second sets of object code. No evidence is provided from either reference to suggest that both a first and a second set of object code are considered in making a selection. Furthermore, the Office Action fails to provide evidence that shows either Tsai or Fuchs teaches first and second sets of object code as claimed.

To further show that the Office Action fails to show the limitations of claim 5, the cited sections of Fuchs and Tsai are provided below:

....

From the cited sections of Fuchs, it appears that Fuchs attempts to recover a faulty process by retrying execution. Tsai appears to restart an erroneous backend with new checkpoint data. Therefore, these cited sections of Fuchs and Tsai make clear that there is no teaching or suggestion of any selecting between first and second object code sets.

*Examiner's response:*

c) Claim 5 recites the limitation "selecting between the first set of object code and the second set of object code in resuming execution of the program", which is interpreted as selecting either one of the object codes in resuming execution. Tsai teaches selecting the second set of object, while Fuchs teaches selecting the first set of object code.

Applicant's argument regarding first and second sets of object code has already been addressed in the Examiner's Response (a) and (b) above. In addition, see the rejection above in paragraph 8 for rejection to claims 5, 10 and 11.

*In the remarks, the applicant argues that:*

d) The alleged motivation states, "It would have been obvious ... to modify the method disclosed by Tsai to include selecting the first set of object code in resuming execution of the program using the teaching of Fuchs [because] one of ordinary skill in the art would be motivated to debug the faulty program, that is, to recover the faulty program."

No explanation is provided nor is it apparent how debugging a faulty program would be provided by including Fuch's progressive recovery algorithm in Tsai's multi-backend system. Furthermore, this alleged motivation is insufficient because it is merely a broad, conclusory statement of a broadly stated function. The alleged motivation lacks clear and particular reasons that would lead one of ordinary skill in the art to combine specific teachings of Fuchs with Tsai. Addressing the "rigorous . .. requirement for a showing of the teaching or motivation to combine prior art references," the Court of Appeals for the Federal Circuit has stated:

...

The alleged motivation is merely a broad conclusory statement of general applicability, and no evidence is provided to suggest the combination. Therefore, the alleged motivation is insufficient to support *prima facie* obviousness.

*Examiner's response:*

d) The modification would be obvious because one of ordinary skill in the art would be motivated to debug the faulty program, that is, to recover the faulty program (Fuchs, column 1, lines 58-67 to column 2, lines 1-32). In addition, see the rejection above in paragraph 8 for rejection to claims 5, 10 and 11.

*In the remarks, the applicant argues that:*

e) The rejection further fails to show that Tsai could be successfully modified with the teachings of Fuchs. For example, Tsai uses modular redundancy to provide fault tolerance (title) apparently because "other conventional schemes use algorithm-based detection methods that are generally not applicable to many types of programs." (col. 1, 11. 61-63). Thus, Tsai appears to teach away from modification using an algorithm such as Fuchs' progressive recovery algorithm. The MPEP states that when a proposed modification would render the teachings being modified unsatisfactory for their intended purpose, there is no suggestion or motivation to make the proposed modification under 35 U.S.C. § 103(a). See MPEP § 2143.01 and In re Gordon, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984) (A § 103 rejection cannot be maintained when the asserted modification undermines the purpose or operation of the main reference.).

*Examiner's response:*

e) Applicant asserts that Tsai "appears to teach away from modification using an algorithm such as Fuchs' progressive recovery algorithm" by merely pointing to a statement in Tsai's column 1, lines 61-63. This statement specifies algorithm-based detection methods. It does not specify progressive recovery algorithm. Therefore, Tsai is not teaching away from Fuchs.

In addition, see the rejection above in paragraph 8 for rejection to claims 5, 10 and 11.

***Conclusion***

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Hilfinger teaches that to invoke the debugger, GDB, the program source code must first be compiled (pg. 28, par. 3, lines 1-14).

McNamara teaches that to invoke the debugger, GDB, the program source code must first be compiled (pg. 1, see section labeled "How do I invoke gdb?").

12. Any inquiry concerning this communication from the examiner should be directed to Qamrun Nahar whose telephone number is (703) 305-7699. The examiner can normally be reached on Mondays through Thursdays from 9:00 AM to 6:30 PM. The examiner can also be reached on alternate Fridays.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki, can be reached on (703) 305-9662. The fax phone number for the organization where this application or processing is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

QN
June 1, 2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100